

Alex BOUTIN
Flavie DUTHAY, Gwenn TANI
Hugo CONAN, Loïk SENLIS, Mathis OLIVIER
Matthieu ROUGER, Tristan RENAUD, Walid HEMAYA

Rapport de conception d'un robot autonome



École polytechnique de l'université de Nantes

Décembre 2019

SOMMAIRE

1. ANALYSE DU CAHIER DES CHARGES	3
2. ÉTAPE DE SPÉCIFICATIONS	7
3. DÉCOMPOSITION FONCTIONNELLE (ET TESTS ASSOCIÉS)	9
4. CONCEPTION DÉTAILLÉE	11
4.1. Le Contrôle Commande	11
4.2. L'Interface Puissance Moteur	16
I - Introduction :	16
II - Le sens de rotation des roues :	17
III - Comment faire varier la vitesse de nos roues?	20
IV - Code du PWM:	25
V - Tests :	26
VI - Conclusion	27
4.3. L'Interface Lampe	28
I - Introduction	28
II - Les photorésistances	28
III - Tests unitaires :	30
IV - Exploitation des résultats	32
V - Application :	33
VI - Conclusion :	36
4.4. L'Interface Labyrinthe	37
I - Introduction	37
II - Télémètre	38
III - Application	41
IV - Conclusion	44
4.5. L'interface de traitement de distances des roues	46
4.6. L'interface de gestion de l'énergie	48
5. ÉTAPE DE RÉALISATION : ASSEMBLAGE DU ROBOT	49
6. CONCLUSION	50
BIBLIOGRAPHIE	51

1. ANALYSE DU CAHIER DES CHARGES

Cahier des charges fourni :

“Il s’agit de réaliser un robot autonome rentrant dans un labyrinthe par l’entrée et sortant le plus rapidement possible par la sortie. La direction de l’entrée et de la sortie est celle d’une source de lumière. Les cloisons représentées ci-dessous sont en bois et ont une hauteur d’une dizaine de cm au maximum et ne masquent pas la source de lumière (il n’y a pas de “toit”). La position de l’ouverture de la cloison intermédiaire n’est pas connue du robot (l’ouverture peut être décalée).”

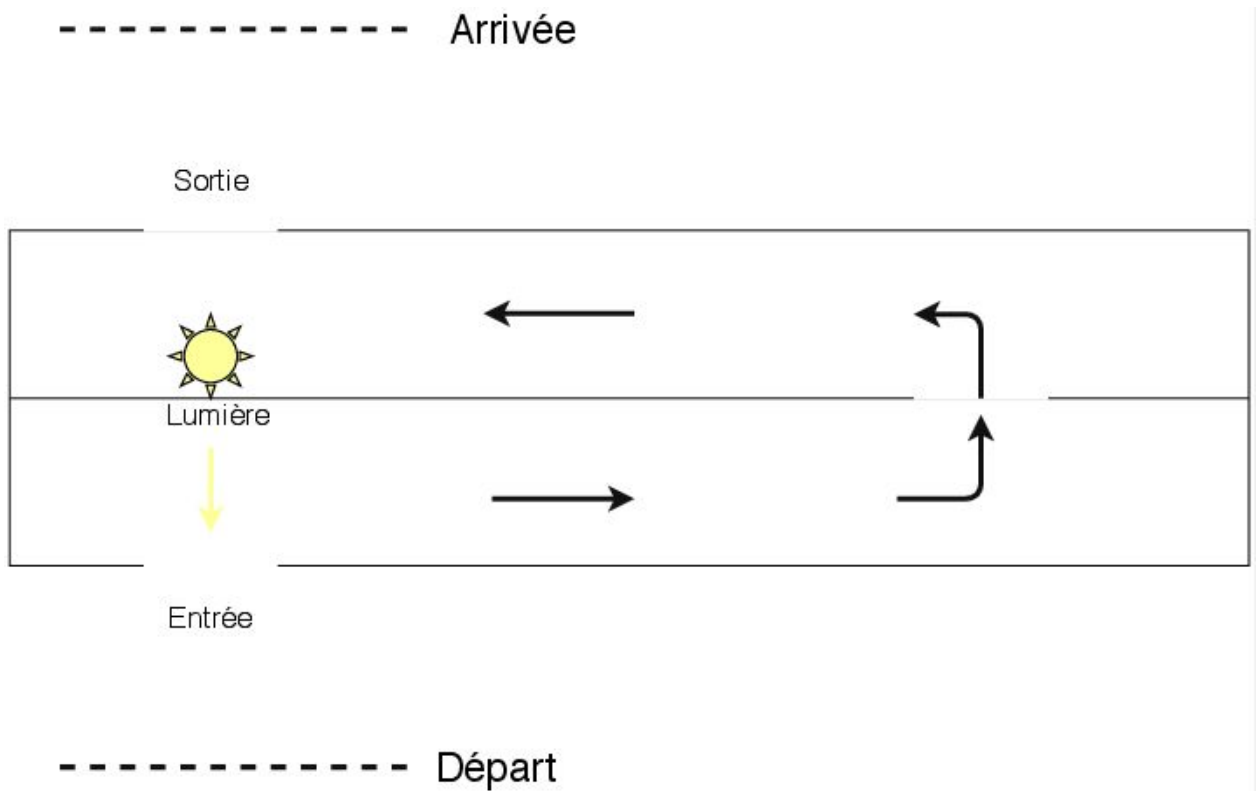


Figure 1 : Schéma du labyrinthe fourni par le cahier des charges

“Deux moteurs électriques et deux roues sont à votre disposition.

La largeur des couloirs et des portes sont de l'ordre de 40-50 cm.

Le concours se déroulera en quatre manches.”

Premièrement, nous nous sommes renseignés auprès du client sur les ressources et le matériel à disposition pour concevoir le robot :

- Plateau du robot 22*22 cm.
- Roue 7 cm de diamètre.
- Moteur (entre 8.5 et 9V).
- Seulement deux roues motorisées (possibilité de rajouter une roue non motorisée).

> Roue non motorisée devant, derrière ?

- Pas de contraintes de poids mais attention à l'équilibre etc ..
- Plateau peut être coupé mais à éviter si possible.

Deuxièmement, le client nous a informé des caractéristiques physiques du labyrinthe que nous devons prendre en compte pour les déplacements de notre robot :

- Parois ayant une hauteur de 10-20 cm.

> En bois avec des trous (qui règlent la position de l'ouverture de la cloison intermédiaire).

- L'entrée fait face à la sortie.
- Sortie et arrivée (distance de l'ordre du mètre) => Ligne virtuelle.
- Nature du sol : horizontal, plat, du lino, quadrillage, marquage interdit.
- Une certaine autonomie car les longueurs des cloison sont non connues.

> Décision à prendre (quand tourner ?).

- Cloison intermédiaire sur la droite (mais distance inconnue).
- Largeur des couloirs comprise entre 40 et 50 cm.
- Parois => aucun marquage, sapin = bois plutôt clair et homogène.
- Pas d'obstacles supplémentaires (juste le labyrinthe).

Troisièmement, nous avons à notre disposition une lampe dont le but est de guider le robot avec certaines exigences :

- Guidé par la lumière => une seule source de lumière qui éclaire l'entrée.
 - Source de lumière en hauteur (30 cm au dessus du sol) très puissante.
- > Donc pas de confusion avec lumière ambiante.

Pour finir, nous sommes soumis à certaines contraintes de déplacement qui sont les suivantes :

- Robot qui doit se déplacer d'un point A à un point B.
 - Déplacement au sein d'un labyrinthe.
 - Quatre manches dans des conditions différentes (un robot à la fois).
- > A chaque fois dans l'axe de l'entrée mais désorienté (angles différents entre normale de la porte et axe du robot).
- Contact contre les parois interdites.
 - Interdiction de franchir les parois par le dessus, ni le dessous.

Ainsi, nous avons extrait les informations nécessaires du cahier des charges, celles propres à la réalisation physique du robot, celles qui nous permettront de déterminer les choses que notre robot devra être capable de réaliser, et celles qui correspondent à l'environnement dans lequel notre robot évoluera. Ces informations nous permettront donc de pouvoir analyser correctement les besoins de notre projet, et d'entamer son étude et sa conception avec tous les éléments nécessaires en main.

Dans le cadre de l'analyse des besoins de notre robot, on déduit donc du cahier des charges que notre robot doit pouvoir se déplacer, à l'aide de deux roues motorisées, mais doit également pouvoir pivoter sur lui même. De plus, il doit être capable de capter une source lumineuse afin de se diriger vers celle-ci. Il doit également détecter les portes du labyrinthe afin de passer par celles-ci sans aucun contact avec les parois.

L'étape de spécification qui suit a pour objectif d'expliciter cette analyse des besoins en étapes simples et concises, sans termes techniques, afin de quantifier les caractéristiques du produit, et les étapes et tests à effectuer pour valider le produit, une fois sa conception terminée.

Nous avons choisi de réaliser ces spécifications sous la forme d'un automate que vous pouvez retrouver à la page suivante.

2. ÉTAPE DE SPÉCIFICATIONS

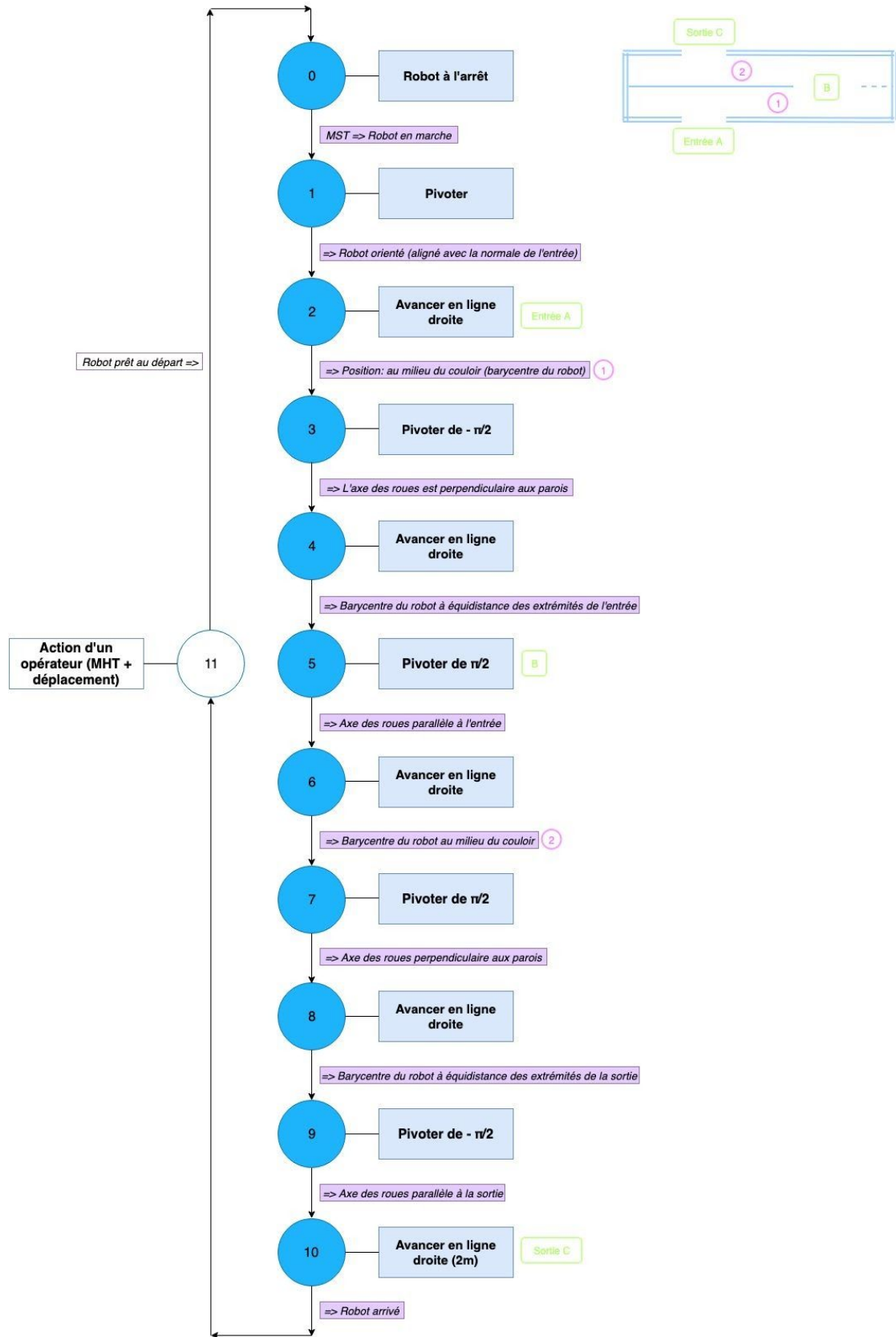


Figure 2 : Automate des spécifications

L'automate ci-dessus de par sa représentation graphique illustre les différentes étapes qui doivent être mise en œuvre de par le contrôle commande, les interfaces lampe, contrôle moteur, contrôle distance et labyrinthe, afin de respecter le cahier des charges.

3. DÉCOMPOSITION FONCTIONNELLE (ET TESTS ASSOCIÉS)

Cette étape du projet est très importante car elle permet de découper ce projet en différents sous-projets, qui seront mieux abordables et réalisables par les membres du projet, qui se décomposent donc en équipe. Dans le cadre de cette conception architecturale, nous avons donc décomposé le travail à réaliser en différentes interfaces :

- Le Contrôle commande
- L'interface Puissance Moteur
- L'interface lampe
- L'interface labyrinthe
- L'interface de traitement de distance des roues
- L'interface de gestion de l'énergie

Nous avons réuni ces différentes interfaces sur le schéma ci-dessous :

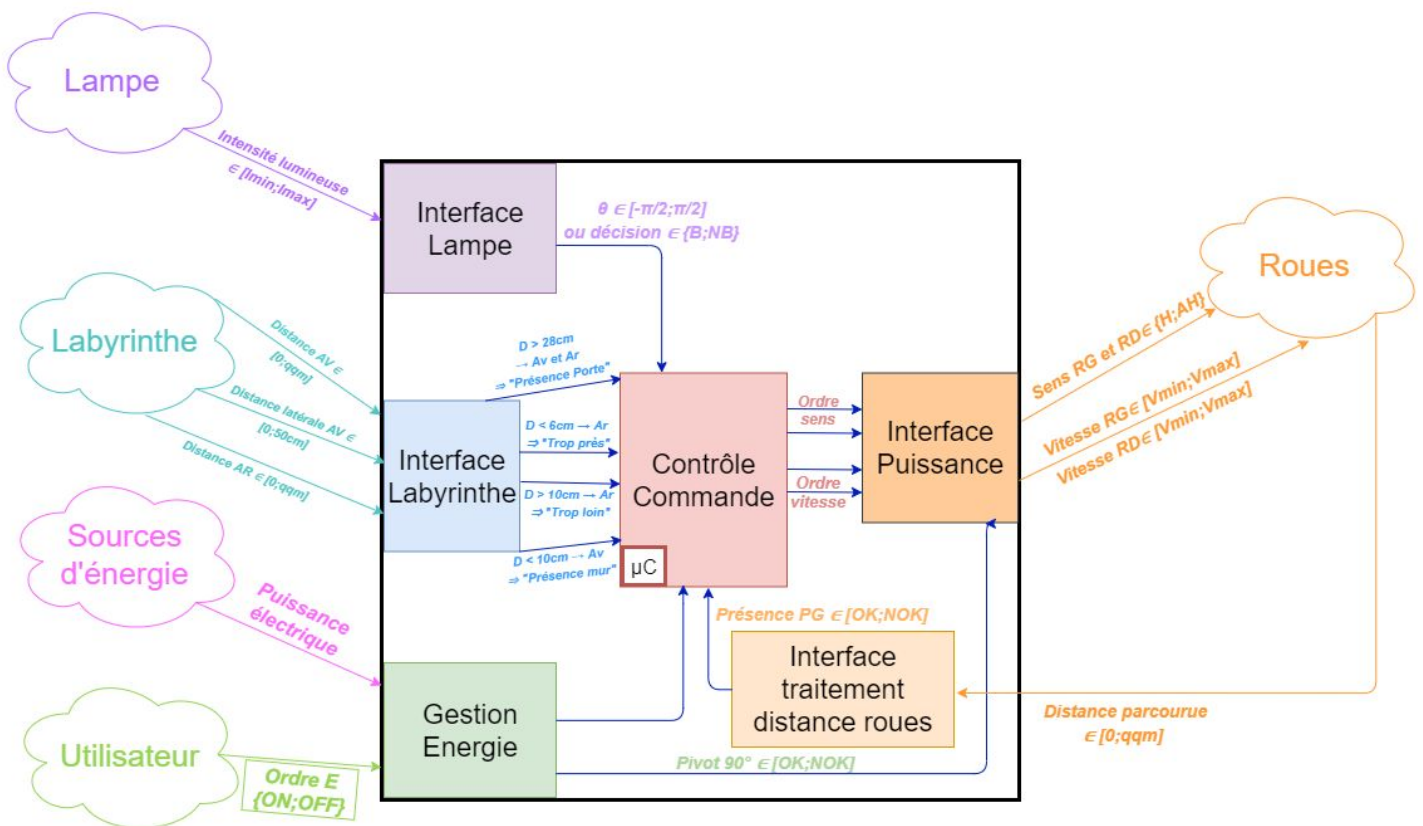


Figure 3 : Décomposition fonctionnelle du projet en différentes interfaces.

Une fois que nous avons réalisé cette décomposition fonctionnelle, nous nous sommes répartis les différentes interfaces à traiter, et nous avons donc commencé l'étude de ces interfaces.

La répartition du traitement des différentes interfaces pour ce projet a été la suivante :

- L'interface lampe a été étudiée et réalisée par Hugo.
- Flavie a travaillé sur l'interface labyrinthe.
- L'interface du traitement de distance des roues a été étudiée et réalisée par Alex.
- Mathis, Matthieu et Gwenn se sont occupés de l'interface puissance moteur.
- Enfin, le contrôle commande étant le plus important car celui-ci est relié à toutes les autres interfaces, Loïk, Tristan et Walid ont travaillé dessus.

La gestion de l'énergie n'est pas demandée dans ce projet.

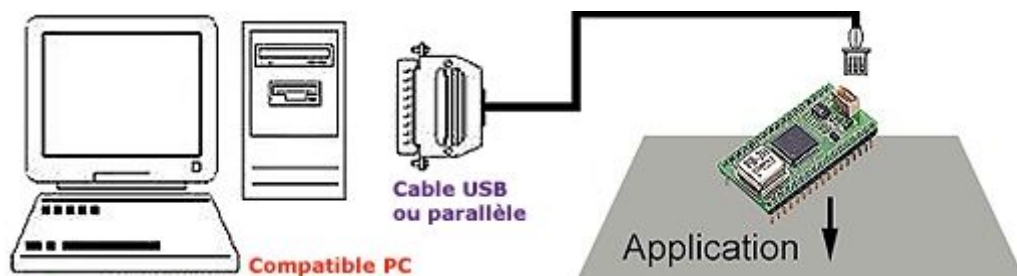
Tous les interfaces correspondent à des sous-projets liés ensemble : en effet, l'interface lampe et l'interface labyrinthe vont envoyer des données sous forme de décision au contrôle commande qui est ici le micro-contrôleur. Ce dernier va envoyer des informations sur le sens et la vitesse des roues à l'interface puissance moteur, en fonction de ce qu'il a reçu précédemment. Enfin, l'interface traitement de distance des roues est liée au déplacement des roues et va renvoyer une information binaire au contrôle commande.

4. CONCEPTION DÉTAILLÉE

4.1. Le Contrôle Commande

Le contrôle commande met en lien toutes les interfaces nécessaires au bon fonctionnement du robot pour respecter le cahier des charges initial. Il utilise donc l'interface lampe, puissance moteur, labyrinthe, distance des roues et traitement de l'énergie.

Pour cela, on utilise le PICBASIC 3B, c'est un petit module qui permet une gestion plus informatisée du signal qui comporte un microprocesseur. Il se programme via un PC avec PICBASIC Studio, programmable en langage BASIC. Le module comporte 28 broches et sa mise sous tension se fait en +5V.



A chaque instant le microprocesseur reçoit en entrée des signaux analogiques qu'il traduit en différents signaux numériques binaires, 1 ou 0. Grâce aux automates disponibles dans chaque partie associée, on peut formuler un programme cohérent avec le cahier des charges.

Ici on peut voir le début d'un programme dans PICBASIC Studio, avec la déclaration des variables et du matériel. Par la suite, les variables et entrées vont effectuer des opérations entre elles. Le programme va se stocker dans une mémoire interne non volatile après compilation du programme. Ainsi, le celui-ci reste disponible après déconnexion du câble.

```

CONST DEVICE = 3B // Définition du modèle du PICBASIC

CONST Decalage = 0 |
CONST Alignement = 7 |
CONST SensG = 8 |
CONST SensD = 15 |
CONST MarcheRG = 10 | // Définition des variable et de leurs port d'entrée/sortie
CONST MarcheRD = 9 |
CONST LBIT1 = 17 |
CONST LBIT2 = 16 |
CONST test = 1 |

DIM B AS Integer |
DIM VRG as Integer | // Signaux interne
DIM R As integer |

GOSUB Marche |
GOTO Lampe | //Appel et renvoi de fonction

```

Figure 5 Exemple de programme en BASIC

En effet le langage Basic est assez simple dans la mesure où il ne nécessite aucune connaissance supplémentaire afin de pouvoir programmer le microcontrôleur.

Pour commencer, on définit le modèle du microcontrôleur qu'on utilise ensuite, en ayant le schéma du Picbasic, nous connaissons toutes les entrées qui vont nous servir pour pouvoir programmer. On définit ensuite toutes les entrées que nous reconnaissons ainsi que toutes les variables qu'on utilisera par la suite.

Tout d'abord, le robot doit pouvoir détecter la plus grande intensité lumineuse et s'aligner avec celle-ci. Sachant que les informations de la photorésistance sont programmés sur 3 bits. Nous avons donc créé:

- Une fonction Accélération, qui entraîne une accélération progressive (théoriquement de l'ordre de quelques centaines de nanosecondes) des roues.
- Des fonctions Avant, Droite, Gauche, qui permettent de définir le sens de rotation de chaque roue dans chaque fonction.

Pour pouvoir détecter la lumière, l'équipe qui travaillait sur l'interface lampe nous a transmis cette table de vérité:

I017	I016	1	0
1		Aller tout droit	Aller à droite
0		Aller à gauche	∅

Table de vérité de l'interface lampe (ici inversée car les I/O du Picbasic sont inversées)

En ayant cette table de vérité, nous avons donc créé une instruction qui s'exécute pour chaque condition comme on peut le voir sur ce code:

Lampe :

```

IF IN(LBIT1)=1 AND IN(LBIT2)=1 THEN
  GOSUB Droit
ELSEIF IN(LBIT1)=1 AND IN(LBIT2)=0 THEN
  GOSUB Droite
ELSEIF IN(LBIT1)=0 AND IN(LBIT2)=1 THEN
  GOSUB Gauche
END IF
IF IN(Tele_avant) = 1 THEN
  GOSUB Arret
  GOSUB Acceleration
  GOSUB ROTATION_D
  GOSUB Droit
  GOTO Labyrinthe
END IF
GOTO Lampe
  
```

Ensuite, une dernière condition a été créée afin que le robot “ne se soucie plus de la lampe” et suive le code du labyrinthe. Cette information est donnée par le télémètre avant. En effet, nous avons donc créé une ligne " If Télé_avant = 1 Then GOSUB Labyrinthe ". Cela a pour effet de sortir du code de la lampe lorsque le robot est aligné avec la lampe et est dans le labyrinthe.

Afin de programmer l’interface labyrinthe, nous avons donc besoin des informations transmises par celle-ci, qui sont envoyées sur quatre entrées.

Sachant que le robot est déjà aligné avec la lampe et est dans le labyrinthe, la première chose à faire était de le pivoter de $\pi/2$, afin qu’il roule dans la bonne direction dans le labyrinthe.

Pour cela, une nouvelle fonction Rotation_D (Droite), à été programmée en précisant alors les informations dont a besoin le robot pour exécuter à bien l’instruction, à savoir l’appel de la fonction Accélération, le sens des deux roues, ainsi qu’une dernière instruction définissant le “ $\pi/2$ ”.

Labyrinthe:

```

IF IN(Tele_pres)=0 AND IN(Tele_loin)=0 THEN      'S'il est bien aligne
    GOSUB Droit
End IF
IF IN(Tele_pres)=1 THEN                          'S'il est trop pres du mur gauche
    GOSUB Dev_D
END IF
IF IN(Tele_loin)=1 THEN                         'S'il est trop loin du mur gauche
    GOSUB Dev_G
END IF
IF IN(Tele_porte)=PresencePorte THEN           'Des qu'il detecte une porte
    GOSUB Arret
    GOSUB Acceleration
    GOSUB Rotation_G

```

MurAVANT:

```

GOSUB Droit
IF IN(Tele_avant)=1 THEN                        'Si le telemetre est a la bonne distance du mur (
    GOSUB Arret
    GOSUB ROTATION_D
    GOSUB Arriere

```

Portefin:

```

IF IN(Tele_porte)=PresencePorte THEN          'Des qu'il detecte une porte
    GOSUB Arret
    GOSUB Acceleration
    GOSUB Rotation_G
    GOSUB Droit
END IF
GOTO Portefin

```

```

END IF
GOTO MurAVANT

```

```

Fin:     END IF
        GOTO Labyrinthe

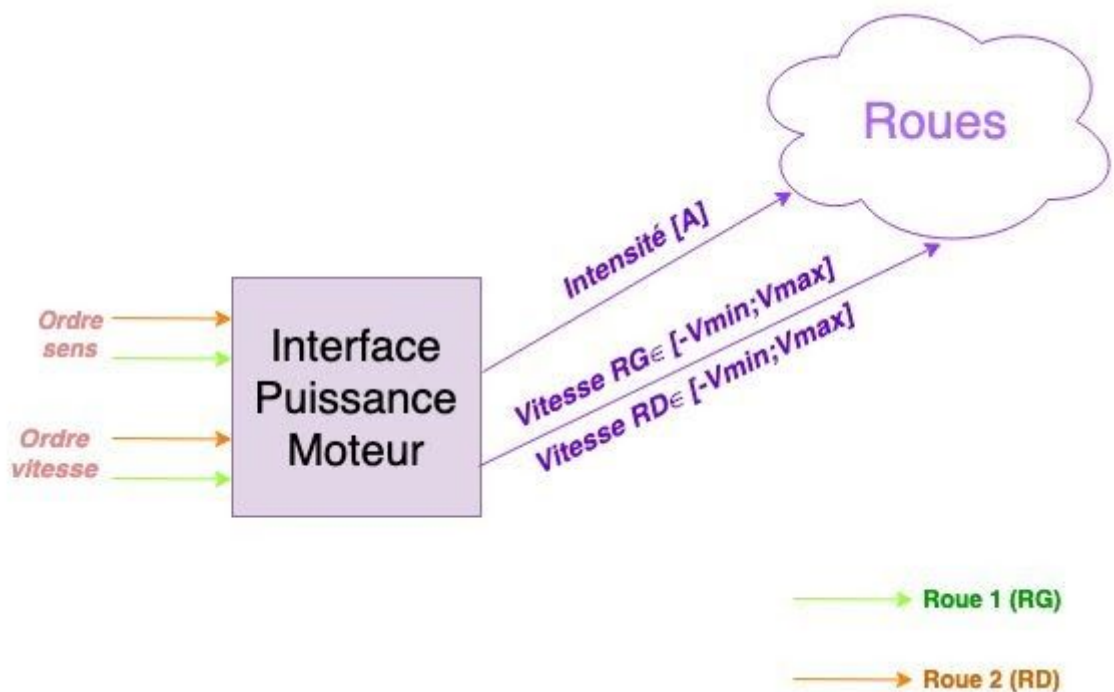
```


4.2. L'Interface Puissance Moteur

I - Introduction :

L'objectif de l'interface puissance moteur, comme son nom l'indique, est de gérer les moteurs liés aux deux roues motrices. Cela implique donc de savoir mettre en fonctionnement ces moteurs, afin de permettre aux roues de tourner, mais cela nécessite aussi de savoir arrêter la rotation des roues quand l'ordre d'arrêt est envoyé par le microcontrôleur. Globalement, l'interface puissance moteur doit savoir donner un sens de rotation aux roues, et doit permettre un arrêt et un démarrage progressif du robot lorsqu'un ordre est donné par le contrôle commande.

Premièrement, nous avons décidé de reprendre la décomposition fonctionnelle de notre interface que nous avons réalisé précédemment avec l'ensemble du groupe afin d'avoir une vision d'ensemble de l'interface que nous devons réaliser.



Interface Puissance Moteur : Représentation schématique

Notre interface reçoit donc quatre informations du contrôle commande: une information de sens et une information de vitesse pour chaque roue.

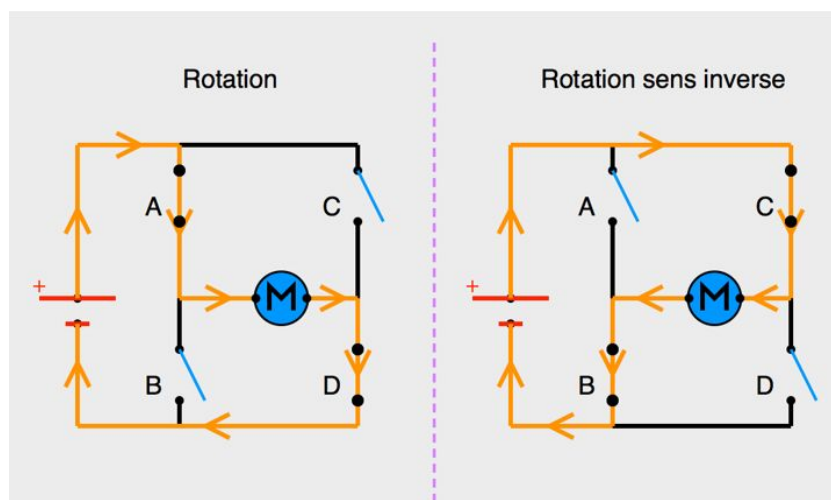
II - Le sens de rotation des roues :

Nous avons commencé par nous intéresser aux informations de sens, car elles paraissent primordiales devant les informations de vitesse.

L'information reçue du contrôle commande pour chaque roue est un bit, c'est à dire un chiffre entre 0 et 1. Le bit 1 représente un sens positif pour le sens de rotation de la roue et 0 un sens négatif. Nous avons donc dû trouver un moyen de transformer chacun de ces chiffres en une tension positive (ici un maximum de 9V), pour le sens positif et une tension négative (minimum de -9V), pour le sens négatif.

Pour cela nous avons choisi d'utiliser un pont en H. Comme son nom l'indique, il correspond à un circuit en forme de H autour du moteur. Chaque patte possède un interrupteur que l'on peut ainsi ouvrir et fermer pour permettre ou non le passage du courant.

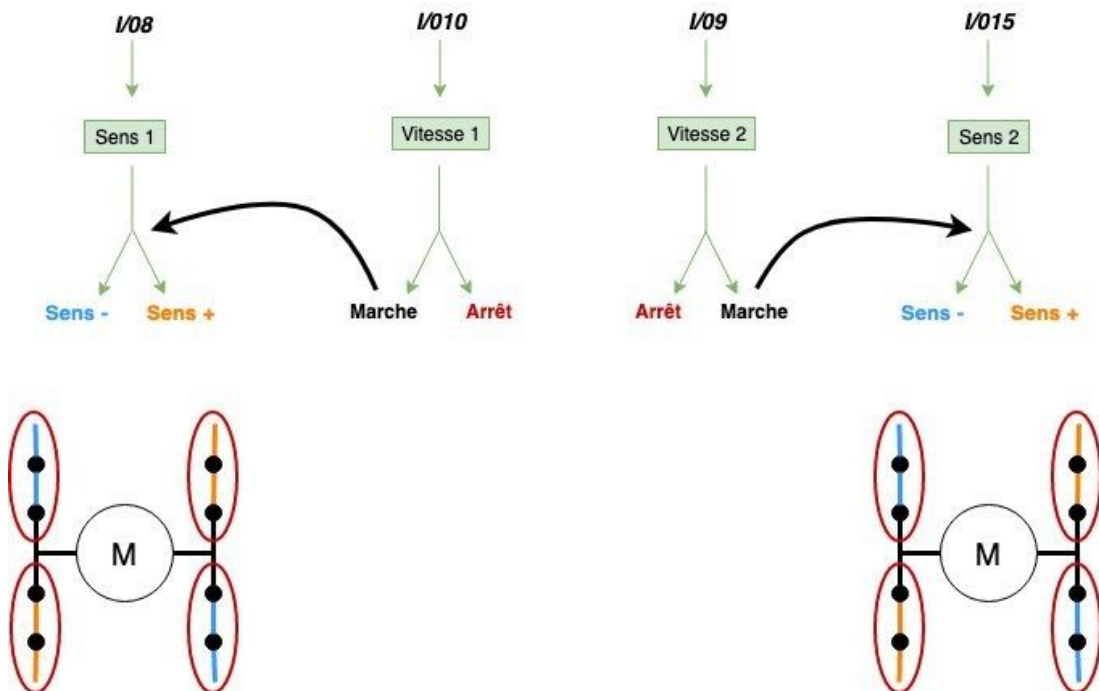
Dans le cas du schéma ci-dessous, nous avons donc quatre interrupteurs, A, B, C, D qui sont liés deux à deux, ainsi, si les interrupteurs A et D sont fermés, le courant entrera dans le moteur par la gauche et sortira par la droite, ce qui assurera la rotation des roues dans le sens avant, tandis que si l'on ferme les interrupteurs B et C, le courant entrera en sens inverse dans le moteur (par la droite) et ainsi, les roues tourneront en sens inverse.



Représentation schématique d'un pont en H - source : <https://openclassrooms.com>

Notre interface puissance moteur fonctionnera donc avec deux ponts en H qui assureront chacun la rotation en sens normal ou inverse de chacune des roues. L'arrivée d'un bit 1 déclenche ainsi l'ouverture des interrupteurs B et C, et l'arrivée d'un bit 0 l'ouverture des interrupteurs A et D.

Voici ci-dessous un schéma récapitulant le fonctionnement des ponts en H et leur intégration à l'interface puissance moteur.



Intégration des ponts en H à l'interface puissance moteur

Ainsi, après avoir compris que nous pouvons utiliser des ponts en H pour gérer le sens de rotation des roues, nous avons cherché un composant parmi ceux à notre disposition qui pourrait remplir ce rôle.

Nous sommes ainsi tombés sur la fiche technique du composant L293D - Quadruple H-drivers, qui comme son nom l'indique est composé de ponts en H.

L293, L293D
QUADRUPLE HALF-H DRIVERS

SLUS090C - SEPTEMBER 1989 - REVISED NOVEMBER 2004

- Featuring Unifrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functionally Similar to QG8 L293 and QG8 L293D
- Output Current 1 A Per Channel (800 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

Description/Ordering Information

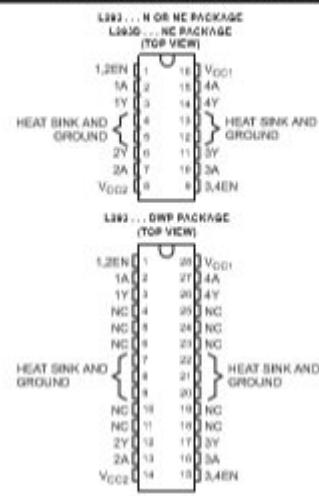
The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 800-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

ORDERING INFORMATION

TA	PACKAGE†	ORDERABLE PART NUMBER	TOP-SIDE MARKING
4°C to 75°C	DIP (DWP)	Tube of 20 L293DWP	L293DWP
	DIP (D)	Tube of 25 L293D	L293D
	DIP (NE)	Tube of 25 L293DNE	L293DNE
	DIP (DE)	Tube of 25 L293DDE	L293DDE

† Package drawings, standard packing quantities, terminal data, symbol definitions, and PCB design guidelines are available at www.ti.com/package.



L293... H OR NE PACKAGE
L293... NE PACKAGE (TOP VIEW)

1,2EN 1 16 V_{CC1}
 1A 2 15 4A
 1Y 3 14 4Y
 HEAT SINK AND GROUND 4 13 HEAT SINK AND GROUND
 5 12
 2Y 6 11 3Y
 2A 7 10 3A
 V_{CC2} 8 9 3,4EN

L293... DIP PACKAGE
(TOP VIEW)

1,2EN 1 26 V_{CC1}
 1A 2 27 4A
 1Y 3 28 4Y
 NC 4 25 NC
 NC 5 24 NC
 NC 6 23 NC
 HEAT SINK AND GROUND 7 22 HEAT SINK AND GROUND
 8 21
 9 20
 NC 19 19 NC
 NC 11 18 NC
 2Y 12 17 3Y
 2A 13 16 3A
 V_{CC2} 14 15 3,4EN

© Copyright © 2004, Texas Instruments Incorporated

Fiche Technique L293D - Fournie pour le projet

Ce composant - qui contient deux ponts en H, l'un sur son côté droit et l'autre sur son côté gauche - est donc idéal pour notre robot, et correspond parfaitement aux contraintes imposées (tension entre 4,5 V et 36 V).

Nous nous sommes intéressés au fonctionnement interne de ce composant et à son intégration dans notre robot plus tard.

La question de sens de rotation des roues étant réglée, nous avons alors une nouvelle problématique : faire varier la vitesse de nos roues. En effet, pour l'ordre vitesse envoyé par le contrôle commande, dans le cas de la réception d'un bit 0, notre moteur devra s'arrêter, et cela, de façon progressive afin de ne pas se stopper brusquement. L'envoi d'un bit 1 en revanche, ordonne la mise en marche du moteur également de manière progressive. Ainsi, nous ne pouvons pas directement envoyer 9V à notre moteur pour le mettre en marche.

Nous nous sommes donc posés la question suivante :

III - Comment faire varier la vitesse de nos roues?

On sait que :

$$E = U - r.I \quad \text{en régime établi, l'intensité } i \text{ est constante}$$

$$\omega = \frac{E}{k} = \frac{U - r.i}{k}$$

On en conclut donc que la vitesse de rotation ω en rad/s dépend de la tension U et de i.

Ainsi pour modifier la vitesse de rotation du moteur, il faut faire varier la force électromotrice (fem) E en agissant sur:

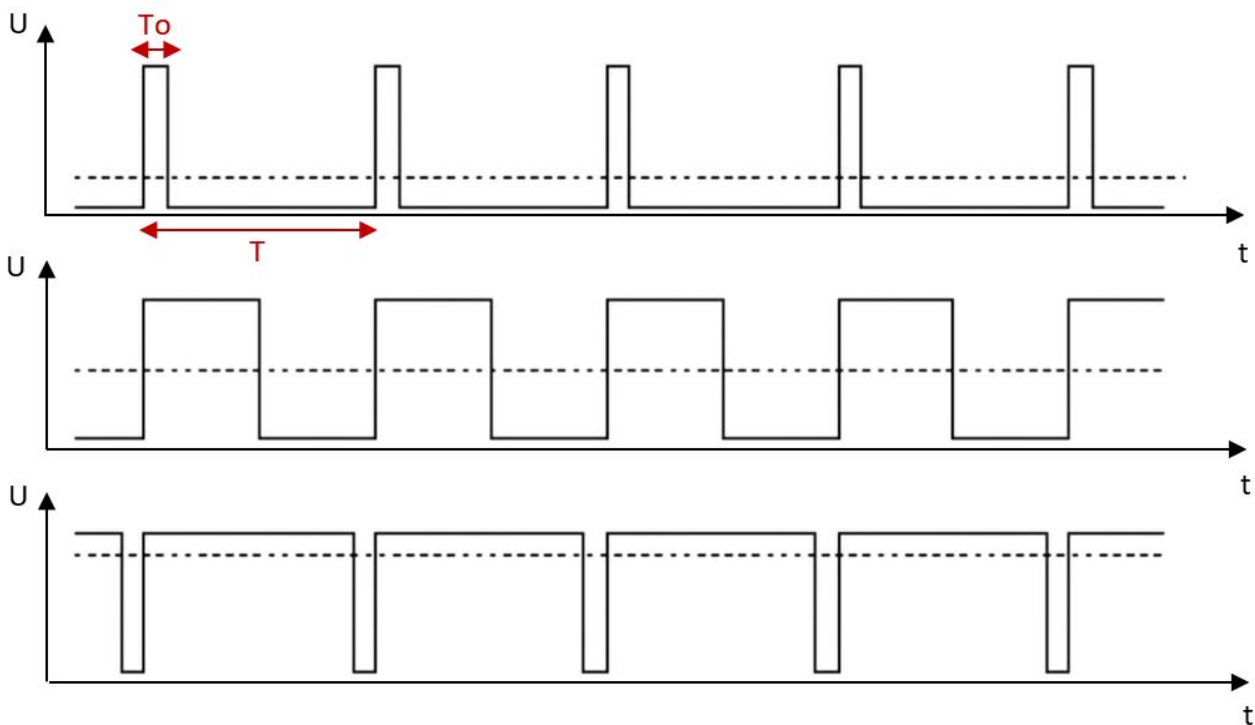
- la tension U (commande en tension)
- le courant I (commande du couple)

Il est important de faire varier la tension envoyée aux moteurs progressivement car passer d'une tension de 0V à 9V instantanément entraînerait un très fort appel de courant risquant une manque d'énergie pour le circuit en entraînant donc une réinitialisation permanente.

Après quelques recherches sur internet, nous avons appris l'existence d'une commande PWM (Pulse with modulation) ou MLI en français (Modulation de Largeur d'Impulsions), qui permet de faire varier la tension envoyée à notre moteur et ainsi modifier la vitesse de nos roues. Nous nous sommes donc intéressés au fonctionnement d'une telle commande.

Nous avons ainsi compris que cette commande consiste à alimenter le moteur avec une tension en créneaux. La tension moyenne reçue par le moteur va alors dépendre alors du rapport cyclique T_0/T . Ainsi la vitesse va pouvoir varier en fonction de cette tension moyenne. En effet, l'augmentation progressive d'états bas de notre tension va entraîner une diminution du rapport cyclique et ainsi diminuer la tension moyenne transmise au moteur et donc diminuer la vitesse de nos roues. Inversement, dans le cas d'une accélération, c'est l'augmentation du nombre d'états hauts qui va augmenter la tension moyenne et permettre l'augmentation de la vitesse de rotation des roues.

Voici un graphique qui représente cette tension en créneau ainsi que la tension moyenne associée :



Tension en créneaux et tension moyenne - <https://www.electronique-mixte.fr/pwm/>

Légendes:

Nous avons ainsi la tension moyenne représentée par la droite en pointillés.

T_0 représente alors la période d'un état haut et T la période correspondant à un état haut et un état bas.

Par la suite nous nous sommes demandés si l'utilisation d'une telle commande était possible dans le cadre de notre projet robot, notamment en terme de fréquence utilisée pour ces signaux PWM.

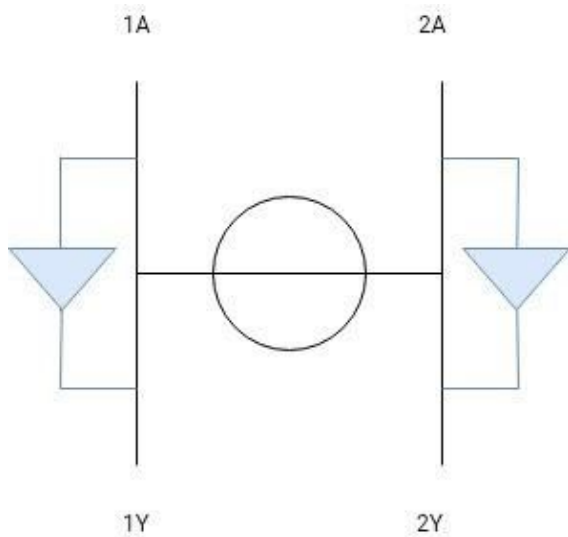
Et en effet, nous avons remarqué qu'il ne faut pas choisir une trop grande fréquence:

- Si la fréquence est trop grande, le signal MLI va devenir trapézoïdal (et non rectangulaire) ainsi la durée des transitions ne sera plus négligeable devant la période et le transistor dissipera davantage d'énergie.

Cependant l'utilisation des signaux PWM correspond parfaitement aux critères pour notre robot et de plus, ils sont parfaitement utilisables en sortie numérique du microcontrôleur PB3B.

Maintenant que nous avons compris comment faire fonctionner notre interface puissance moteur et gérer la vitesse et le sens de rotation de nos roues, nous nous sommes intéressés au fonctionnement du composant L293D ainsi qu'à la façon de l'intégrer à notre circuit.

De plus, un inverseur est placé au sein même du pont en H pour transmettre l'information aux branches de sortie du pont en H de la façon suivante :



Ainsi à partir d'un unique bit d'information envoyé par le microcontrôleur, nous sommes capable de transmettre quatre informations différentes aux quatres branches de notre pont en H afin de gérer l'ouverture ou la fermeture des interrupteurs.

Schéma d'un pont en H

Le composant étant déjà intégré au circuit nous n'avons plus qu'à effectuer les branchements nécessaires liant les moteurs au reste du circuit et permettant ainsi la transmission de l'information.

Pour les ondes PWM, nous remarquons que les bits d'information viennent de I/O9 et I/O10 jusqu'aux bornes ENABLE 1,2 et ENABLE 3,4 de notre composant L293D.

Il ne nous reste plus qu'à implémenter le code nécessaire au fonctionnement des ondes PWM sur Pic Basic et notre interface puissance moteur sera parfaitement en état de marche.

IV - Code du PWM:

Le programme à implémenter sur la carte PicBasic est assez simple. En effet, en cherchant sur le manuel PicBasic on trouve qu'il existe une commande permettant de créer un signal PWM (signal rectangulaire de fréquence fixe mais avec un rapport cyclique variable) qui va permettre de contrôler la vitesse des moteurs et donc des roues.

Pour mettre en marche les moteurs, il suffit de générer un signal PWM sur les ports 9 et 10 (ou \09 et \010) du PicBasic-3B en faisant varier le rapport cyclique plus ou moins rapidement pour que la vitesse des roues augmente plus ou moins rapidement.

Au niveau du programme dans PicBasicStudio, nous avons créé la fonction "Accélération" suivante :

```
CONST MarcheRG = 10
CONST MarcheRD = 9
Acceleration:FOR Vitesse = 0 TO 255
    PWM MarcheRG,Vitesse
    VitesseRD = Vitesse
    IF VitesseRD > 180 THEN
        VitesseRD = 180
    END IF
    PWM MarcheRD,VitesseRD
NEXT
RETURN
```

Les ordres donnés à la roue droite sont associés au port \09.

Les ordres donnés à la roue gauche sont associés au port \010.

Le variable "vitesse" va augmenter progressivement ce qui va augmenter progressivement la vitesse des 2 roues.

Ensuite, pour arrêter les roues, il suffit de "désactiver" les signaux PWM, c'est-à-dire stopper leur génération en utilisant l'instruction "PWMOFF" comme dans l'exemple ci-dessous :

EXEMPLE

```
10      PWM 9,191
20      DELAY 255
30      PWMOFF 9          ' Stoppe le signal PWM
```

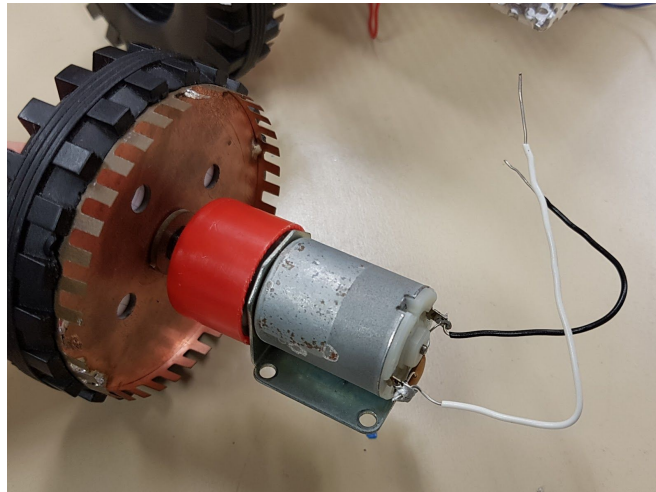
Cependant, pour éviter un très fort appel de courant entraînant un manque d'énergie pour le circuit, on va éviter d'utiliser "PWMOFF" pour ne pas réinitialiser la tension.

La tension ne va donc pas passer instantanément de 0 à 9V mais augmenter ou diminuer progressivement. Pour cela, on crée une fonction "Décélération" ressemblant à la fonction "Accélération" mais en faisant varier la variable vitesse de 255 à 0.

```
Deceleration:FOR Vitesse = 255 TO 0
              PWM MarcheRG,Vitesse
              IF Vitesse < 75 THEN
                  Vitesse = 75
              END IF
              VitesseRD = Vitesse - 75
              PWM MarcheRD,VitesseRD
NEXT
RETURN
```

V - Tests :

Le matériel utilisé pour l'interface puissance moteur de notre robot sont deux moteurs "GMH04" de la marque Lynxmotion, associé à des roues comme le montre l'image ci-dessous :



Tout d'abord, il a fallu réfléchir au sens dans lequel devrait tourner les moteurs (avec les roues) pour tourner à gauche, à droite ou aller tout droit.

Pour aller tout droit, le moteur gauche doit tourner vers la gauche et le moteur droit doit tourner vers la droite (en regardant face aux roues).

Pour tourner à gauche, les deux moteurs doivent tourner vers la gauche et pour tourner à droite, les deux moteurs doivent tourner vers la droite.

On peut remarquer sur la photo au dessus que les moteurs possèdent 2 fils. Il a ensuite fallu vérifier que les moteurs fonctionnaient bien. Nous avons donc simplement branché les fils des moteurs au même générateur en faisant varier la tension fournie pour observer le comportement des moteurs. Nous avons rapidement observé une vitesse légèrement différente pour l'un des deux moteurs ce qui peut entraîner une petite déviation du robot quand celui-ci est en marche.

Pour régler ce problème, nous avons pensé à modifier l'un des signaux PWM pour augmenter la vitesse de rotation du moteur plus lent, Nous aurions aussi pu utiliser les télémètres de l'interface labyrinthe qui aurait permis au robot de se remettre droit en cas de déviation trop importante.

Nous avons enfin réalisé un test d'intégration en reliant les moteurs à la carte PicBasic pour tester de commander la vitesse et le sens des roues en codant des instructions sur PicBasicStudio.

VI - Conclusion

L'interface de puissance moteur comprend donc des ponts en H permettant de faire varier le sens de rotation des moteurs associés aux roues, des signaux PWM générés par le PicBasic utilisés pour faire varier la vitesse des roues. La vitesse et le sens de rotation des moteurs peuvent facilement être contrôlés par des instructions écrites sur PicBasicStudio.

4.3. L'Interface Lampe

I - Introduction

Le but de l'interface lampe est de permettre au robot de s'orienter par rapport à la normale de l'entrée du labyrinthe. Pour se faire il y a une lampe située derrière l'entrée du labyrinthe permettant de repérer l'entrée. Une fois aligné, le robot pourra avancer vers l'avant afin d'entrer dans le labyrinthe.

D'après le cahier des charges le robot sera placé à 1m50 de l'entrée du labyrinthe et formera un angle compris entre -60° et 60° avec la normale de l'entrée du labyrinthe. Le cahier des charges ne le spécifie pas mais il est évident que la trajectoire du robot devra être ajustée au fur et à mesure si il dévie. Ainsi le but de l'interface lampe évolue et est désormais d'envoyer en continue au contrôle commande une information du type "aligné" ou "non aligné". Cette information permet de déterminer si le robot doit avancer ou tourner.

Afin de remplir le cahier des charges, il nous fallait donc trouver un moyen de traduire l'intensité lumineuse émise par la lampe en information pour le contrôle commande. Sachant que le contrôle commande est constitué d'un microcontrôleur PICBASIC 3B ne pouvant recevoir les informations que sous forme binaire, niveau haut et niveau bas de tension dans notre cas. Ainsi nous avons pensé à transformer l'information intensité de la lampe en information tension pour le contrôle commande. Pour réaliser cette transformation nous avons pensé à utiliser des photorésistances afin d'obtenir une valeur de résistance, nous permettant ainsi d'obtenir indirectement une tension.

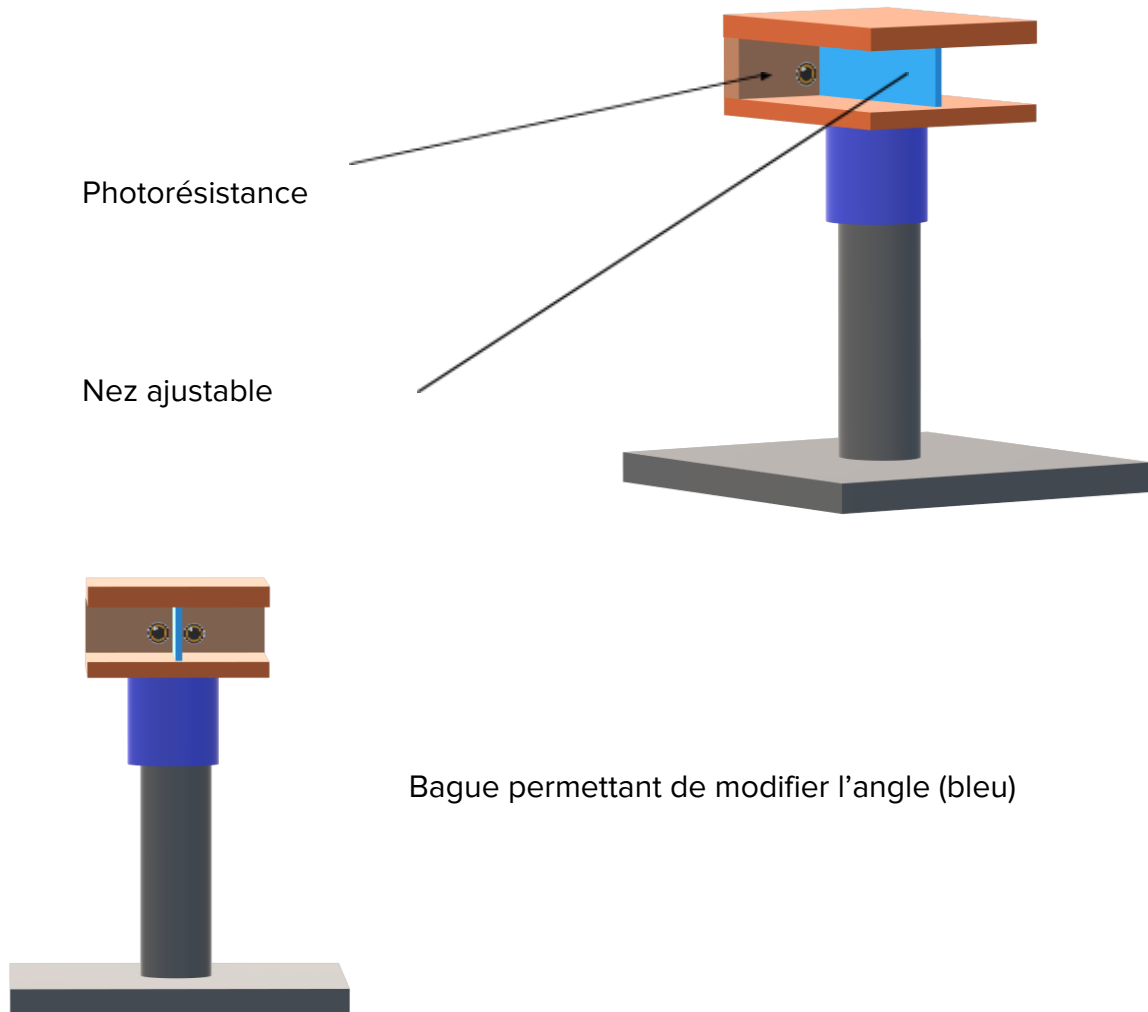
II - Les photorésistances

Une photorésistance est une résistance dont la valeur varie en fonction de l'intensité lumineuse reçue. Ce composant nous permet de mesurer l'intensité lumineuse, par le biais de la résistance mesurée. Sachant l'intensité lumineuse nous pourrions savoir si le robot est aligné avec la lampe ou non.

Cependant, nous ne savions pas comment les utiliser ni même combien en utiliser. Passé cette étape un appareil de mesure (schéma 1) nous a été fournis. Cet appareil se constitue de deux photorésistances montées en série, elles sont séparées par une planche que l'on appellera le "nez" par la suite. Nous mesurons la résistance des photorésistances à l'aide d'un ohmmètre. Le

nez est ajustable en longueur et l'angle que forme les photorésistances avec la normale du plan de la lampe modifiable.

Schémas et modélisations de l'appareil de mesure utilisé :



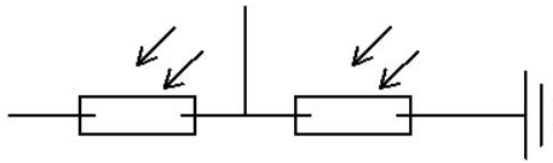
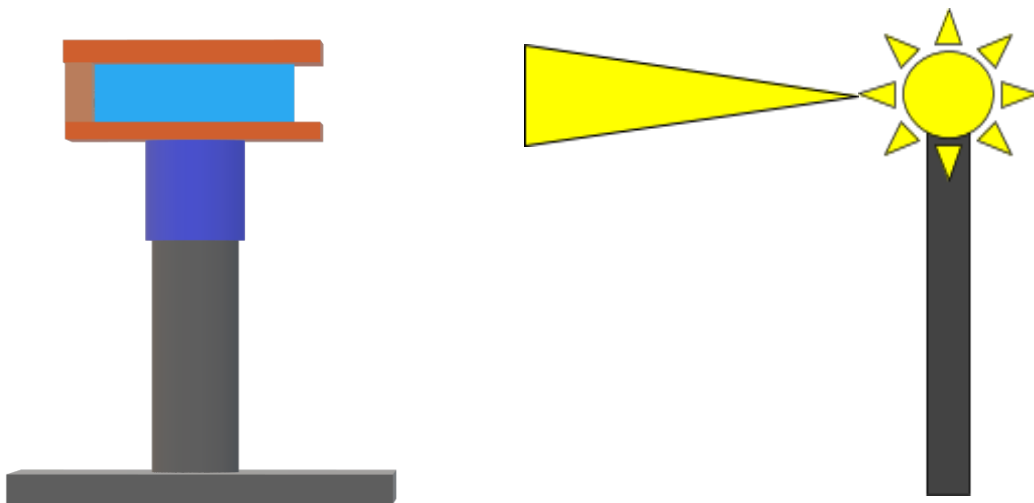


Schéma électrique des deux photorésistances



III - Tests unitaires :

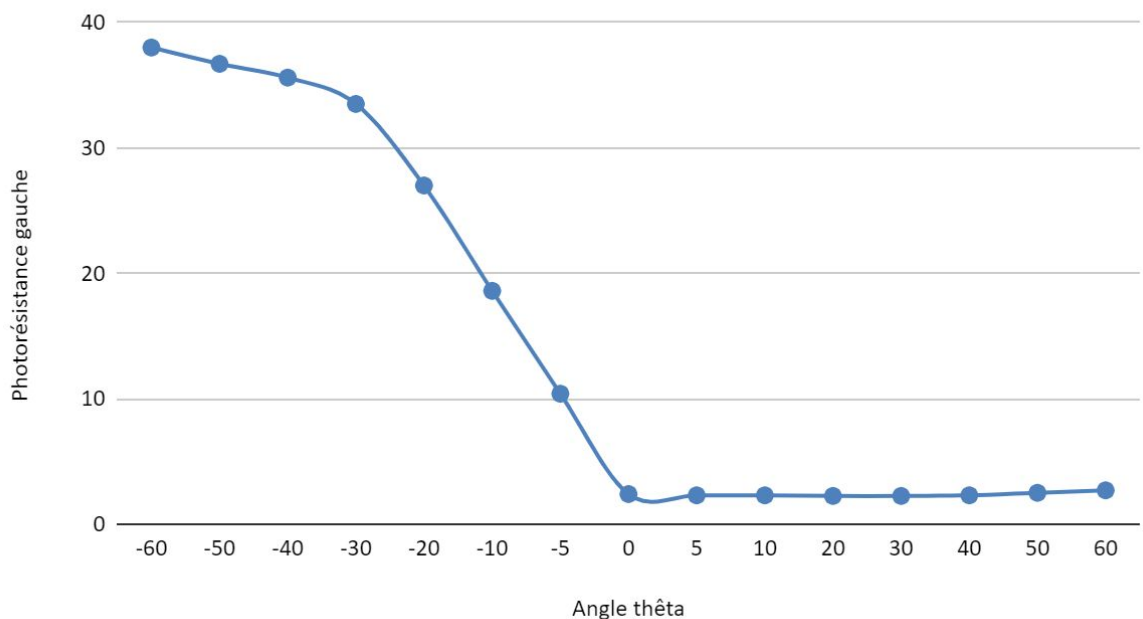
Nous avons donc effectué des mesures afin de comprendre l'évolution de la résistance fournie par la photorésistance en fonction de l'angle formée avec la normale du plan de la lampe. Nous remarquons très vite avec de premières mesures approximatives pour comprendre grossièrement le fonctionnement des photorésistances que :

- Les résistances fournies par les photorésistances diminuent si l'intensité reçue par les photorésistances augmente. (graphique 1)

- Les photorésistances sont très sensibles à l'ombre (graphique 1). Il y a une grosse différence de valeurs entre les valeurs obtenues lorsque les photorésistances sont éclairées directement ou non.
- Les résistances fournies par les photorésistances droite et gauche sont symétriques. En effet, lorsque l'une des photorésistance augmente, l'autre diminue et inversement. Les valeurs sont semblables dans les deux cas.

Pour savoir si le robot est aligné ou non il nous faut un point de comparaison c'est pour cela que nous avons choisi d'utiliser deux photorésistances. De plus, en utiliser deux nous permettrait d'obtenir l'information "trop à gauche" et "trop à droite" en plus de l'information "aligné". Afin d'obtenir des résultats plus précis et plus facilement exploitables nous avons effectué les mesures pour les résistances droites et gauches.

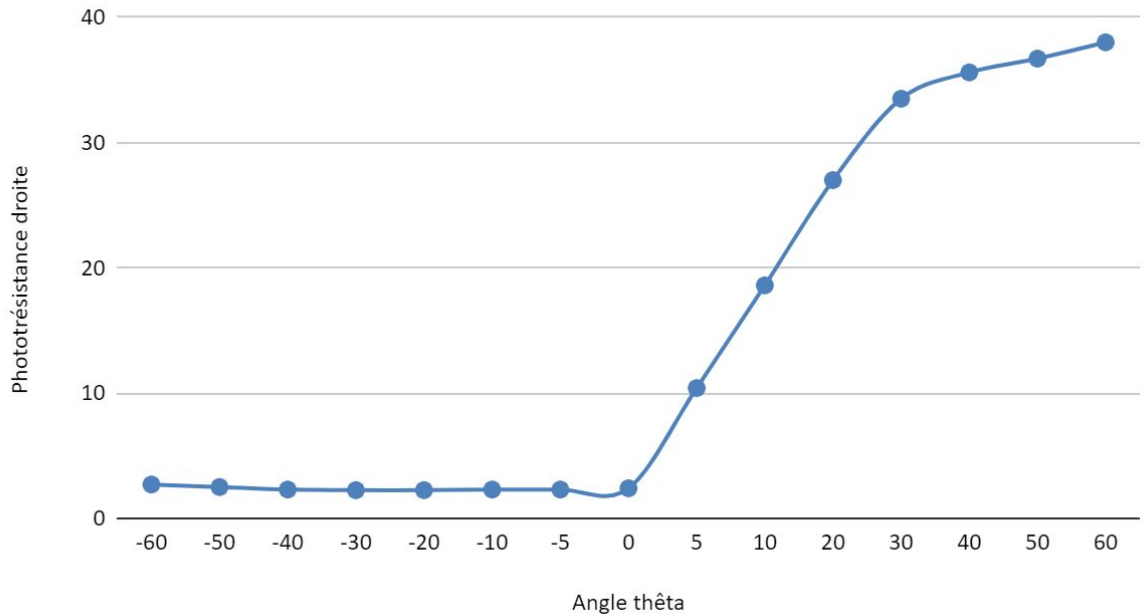
graphique 1 : Evolution de la résistance gauche en fonction de l'angle θ



Nous avons observé précédemment un axe de symétrie, ainsi nous avons considéré les valeurs des résistances droites et gauches identiques.

Nous obtenons donc un second graphique symétrique au premier :

graphique 2 : Evolution de la résistance droite en fonction de l'angle θ



IV - Exploitation des résultats

Une fois ces valeurs de résistances obtenues nous avons cherché à obtenir une valeur de tension en fonction de l'angle θ .

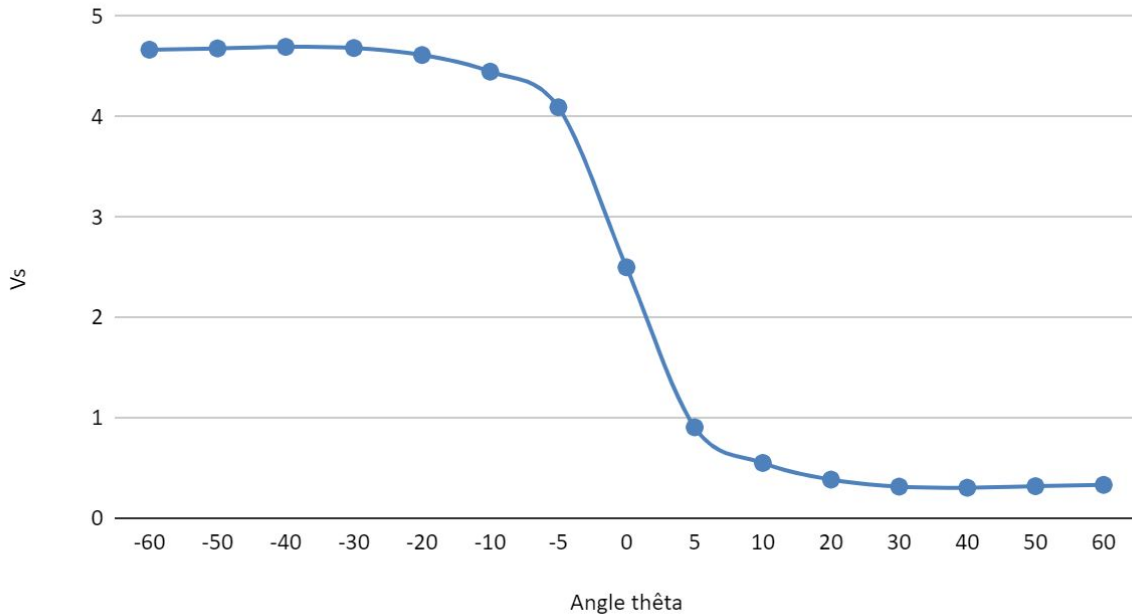
Les photorésistances, telles qu'elles sont montées, se comportent en pont diviseur de tension et fonctionnent en push-pull d'une certaine manière (Quand une résistance augmente, l'autre diminue).. Ainsi, sachant que notre tension d'entrée pour les photorésistances est de 5V, on peut obtenir une tension de sortie qui serait le résultat de la formule du pont diviseur de tension suivante :

$$V_s = V_e \frac{R_1}{R_1 + R_2}$$

Ici, R1 est la résistance gauche et R2 la résistance droite. Ve est égal à 5V.

De cette formule nous avons pu tracer l'évolution de la tension de sortie en fonction de l'angle nous permettant d'avoir l'angle en fonction de la tension (graphique 3).

graphique 3 : Evolution de la tension de sortie en fonction de l'angle θ



Une fois cette fonction obtenue nous cherchons à déterminer quel est l'angle limite à partir duquel le robot rentre dans le labyrinthe sans encombre avec les données du problème (distance de départ, ouverture de l'entrée, taille du robot...). Suite à nos calculs l'angle seuil est de 4.6° .

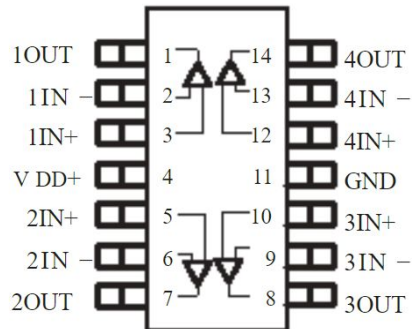
V - Application :

De ce fait, grâce à l'angle de 4.6° degré que nous avons trouvé précédemment, nous en déduisons que nos valeurs limites de tensions à partir desquelles le robot sera considéré "aligné" par le contrôle commande sont $U_1=3.7\text{ V}$ & $U_2=1.2\text{ V}$.

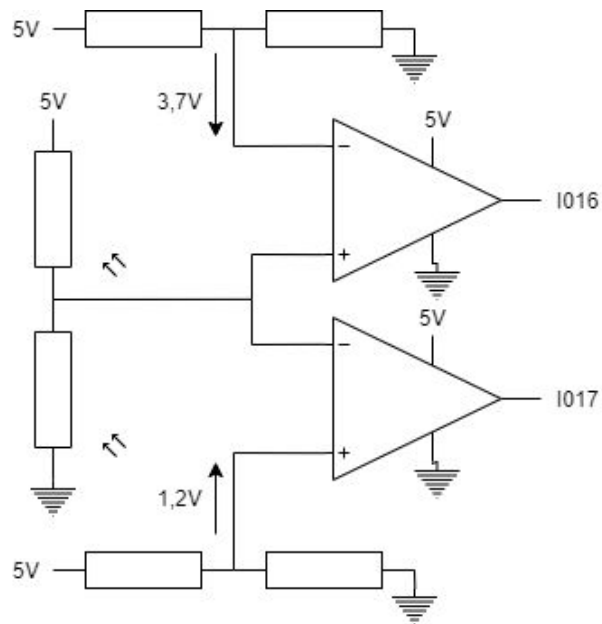
Il faut désormais pouvoir traduire ces tensions seuils afin de retourner l'information binaire que le contrôle commande va consommer. Cette information sera codée sur 2 bits permettant ainsi de renvoyer 4 informations :

- Aligné
- Trop à gauche
- Trop à droite
- inexistant

Pour ce faire, nous utilisons des comparateurs de tensions grâce à un amplificateur opérationnels TLV 2464 CN tel que celui-ci :



Les comparateurs de tensions auront pour but de comparer la tension renvoyée par les photorésistances avec les tensions seuils obtenues précédemment. Nous avons donc construit le circuit suivant :



Ainsi, les amplificateurs opérationnels ont permis d'obtenir une information binaire en niveaux hauts et niveaux bas, le tout codé sur 4 bits. Il nous reste à déterminer la table de vérité de nos résultats afin que le PICBASIC puisse traduire l'information et prendre une décision.

I017	I016	0	1
0		Aller tout droit	Aller à droite
1		Aller à gauche	∅

Table de vérité de l'interface lampe

VI - Conclusion :

Il ne reste plus qu'au contrôle commande qu'à consommer l'information que lui donne l'interface lampe afin de pouvoir prendre une décision.

Lors de la réalisation de cette interface nous avons permis à l'information intensité lumineuse transmise par la lampe de devenir une information binaire pour le contrôle commande.

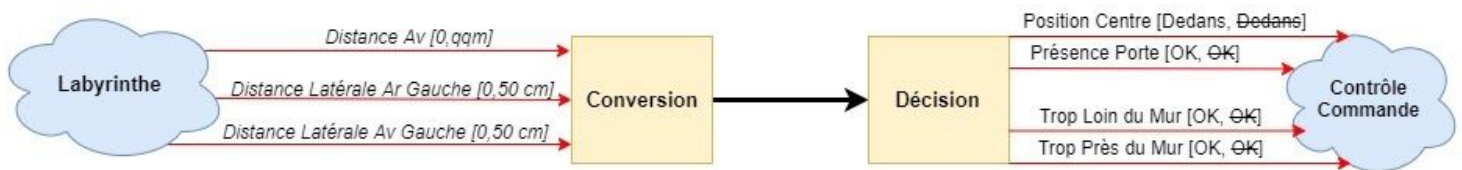
Nous avons rencontré des difficultés, notamment pour comprendre que les résistances agissaient comme un pont résistif et permettent ainsi de transformer l'information intensité lumineuse en information tension.

4.4. L'Interface Labyrinthe

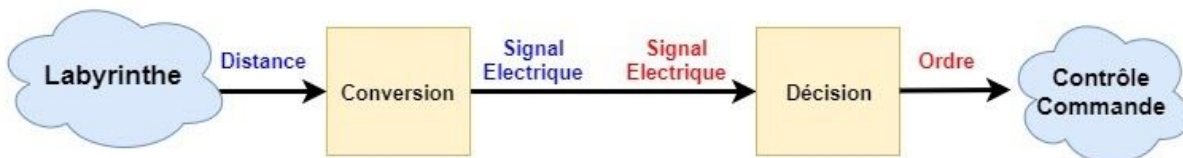
I - Introduction

Le principe de l'interface labyrinthe est de récupérer des distances afin d'envoyer les décisions correspondantes au microcontrôleur. En effet, le robot autonome doit pouvoir se diriger dans le labyrinthe et d'après le cahier des charges sans toucher les murs. Il faut que le robot avance jusqu'à une distance limite du mur et que celui-ci s'arrête pour suivre la suite du trajet.

Nous allons devoir réfléchir à la manière de transformer une distance en décision. La difficulté se trouve ici. Cependant, nous pouvons convertir notre distance en une autre donnée, afin que cette donnée permette de prendre une décision.



Heureusement pour nous, nous savons qu'à partir d'une tension, une décision peut être prise. C'est notamment le cas à partir d'un comparateur de tension. Nous avons, ainsi, résolu une partie du problème.



Il nous reste à trouver un composant pouvant convertir une distance en tension. Il se trouve que le télémètre est un appareil permettant de faire ceci.

Ainsi, pour le moment, nous avons un télémètre qui va nous renvoyer une tension en fonction de la distance qu'il mesure jusqu'à un obstacle. La tension du télémètre est alors envoyée dans un comparateur de tension.

Cependant, il faut que l'on puisse définir une tension seuil qui correspondra à la distance limite de notre choix. Un pont diviseur de tension nous permettra de le faire. Une tension seuil sera envoyée dans un comparateur de tension afin de servir de référentiel pour la prise de décision.

A noter que le cahier des charges stipule que le fonctionnement ce circuit sera sur du 5V. Nous devons faire attention à cette information dans le choix de nos composants.

II - Télémètre

Nous avons à notre disposition le télémètre SHARP 2D120. D'après la datasheet de ce dispositif, celui-ci fonctionne entre une tension de 4,5 V et 5,5V. Nous sommes sûrs que l'appareil sera bien alimenté. De plus, le domaine de fonctionnement du télémètre est entre 4cm et 30cm. Ces dimensions sont plus que suffisantes pour être utilisées dans à l'échelle de notre labyrinthe. Autre information très importante récupérée dans la datasheet, pour un bon fonctionnement le télémètre doit être positionné en position verticale.

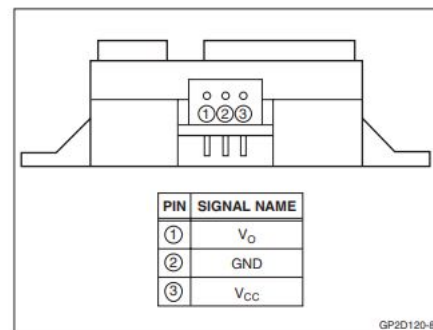
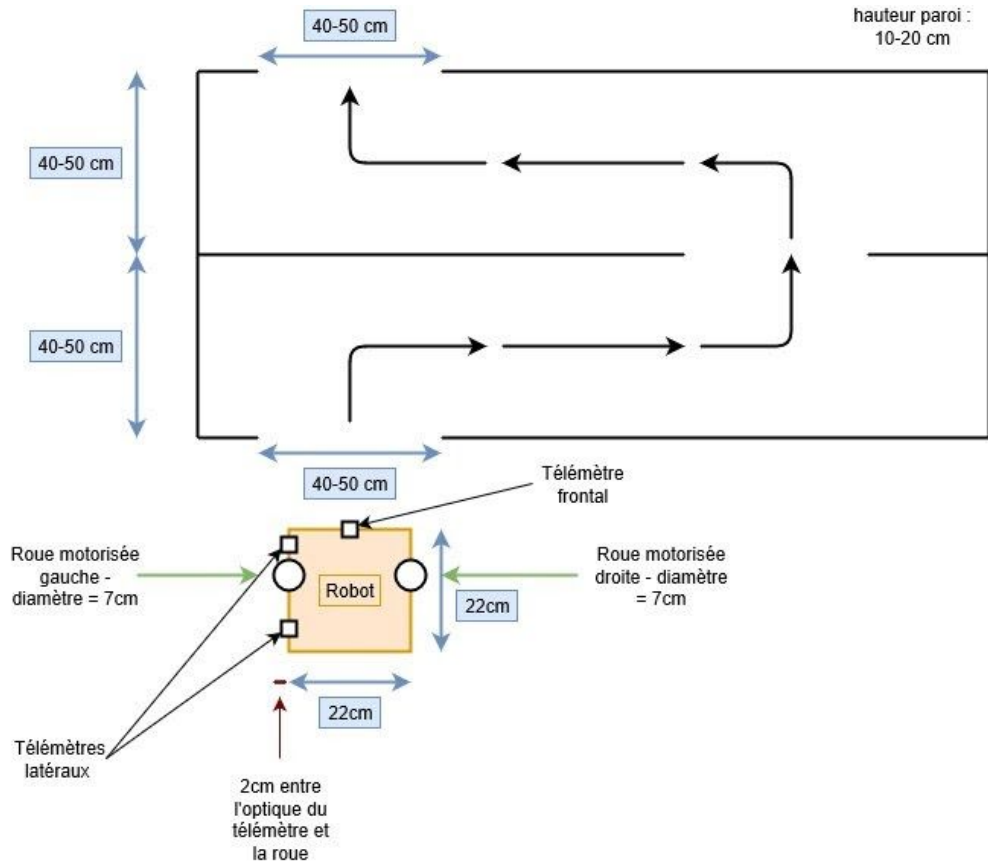


Figure 1. Pinout

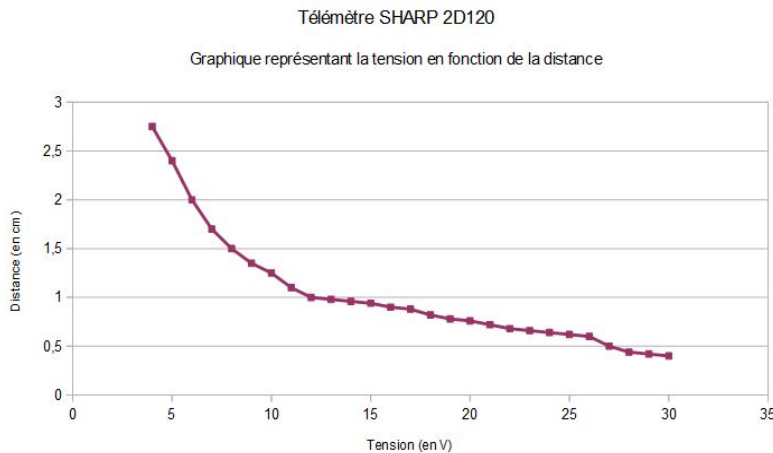
L'objectif est de savoir combien de télémètre utiliser. Prenons le schéma du labyrinthe ci-dessous. Le robot doit être capable de détecter un obstacle devant lui, ainsi un télémètre doit être mis à l'avant du robot. De plus, le robot doit être capable de détecter une porte sur sa gauche, le groupe a décidé, de mettre 2 télémètres du côté gauche du robot afin de la détecter. L'utilisation de 2 télémètres du même côté est directement lié au choix de faire rouler le robot en marche après la deuxième porte.

Ainsi, l'utilisation de 3 télémètres a été nécessaire lors de la conception du robot.

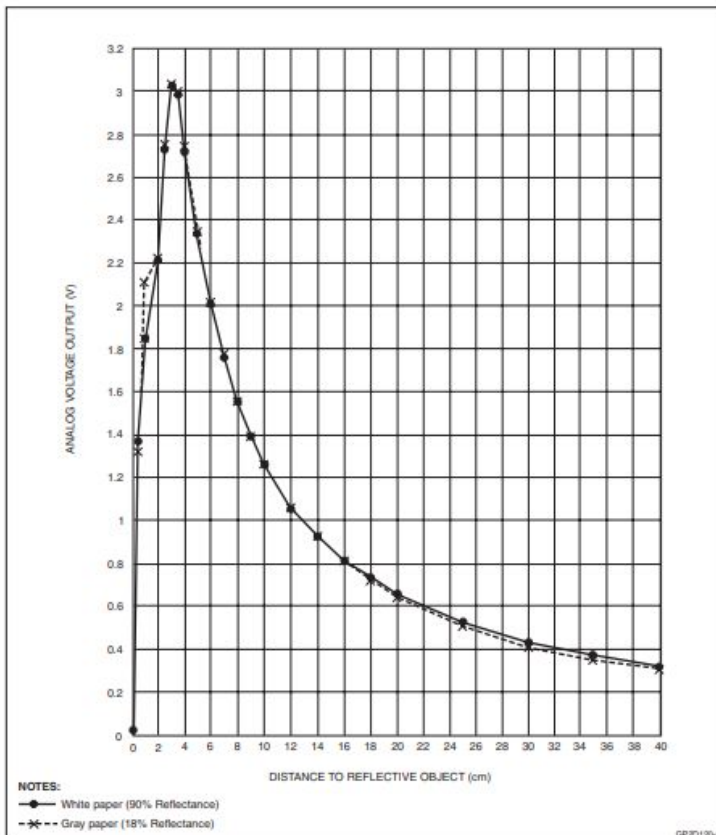


III - Test unitaire

Avant d'utiliser nos télémètres, il y a quelques précautions à prendre. Il faut effectuer des tests unitaires sur les télémètres, afin de vérifier si ceux-ci vérifient la courbe de la datasheet. Etant donné que la datasheet stipulait que le télémètre ne pouvait fonctionner seulement entre 4cm et 30cm, nous avons fait une mesure de tension correspondante tous les 1cm.



Ces mesures nous ont permis de tracer à notre tour une courbe. Nous pouvons constater que notre télémètre correspond plutôt bien. On retrouve sur les deux courbes une chute rapide de la tension entre 4cm et 15cm et un ralentissement juste après. De plus, le maximum et le minimum des deux courbes sont semblables.



Distance (cm)	Tension (V)
4	2,75
5	2,4
6	2
7	1,7
8	1,5
9	1,35
10	1,25
11	1,1
12	1
13	0,98
14	0,96
15	0,94
16	0,9
17	0,88
18	0,82
19	0,78
20	0,76
21	0,72
22	0,68
23	0,66
24	0,64
25	0,62
26	0,6
27	0,5
28	0,44
29	0,42
30	0,4

Le tableau des tensions associées aux distances nous sera très utile pour le choix des seuils de tension.

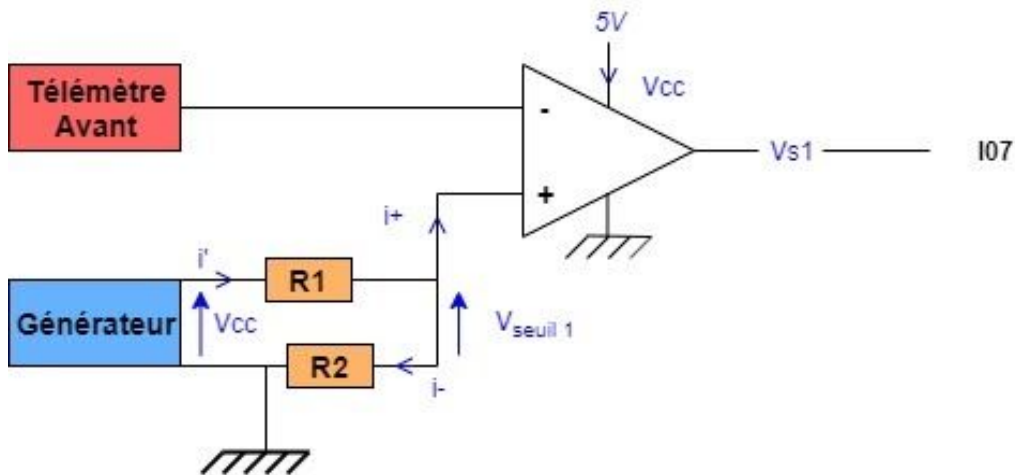
IV - Application

Pour le télémètre frontal, il a été décidé que la distance limite la plus efficace serait 10cm en prenant en compte la taille du robot et la largeur du couloir. Ainsi, $V_{seuil1} = 1,25\text{ V}$, d'après le tableau construit à partir du test unitaire pour le pont diviseur de tension.

Dans cette configuration lorsque

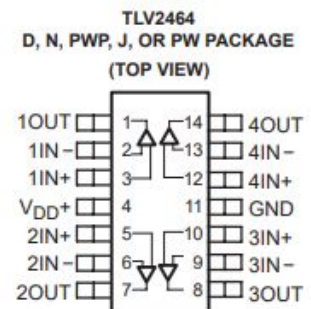
$V_{seuil1} > V_T \Rightarrow 5\text{V}$ (Niveau Haut) / Absence de mur

$V_{seuil1} < V_T \Rightarrow 0\text{V}$ (Niveau Bas) / Présence de mur

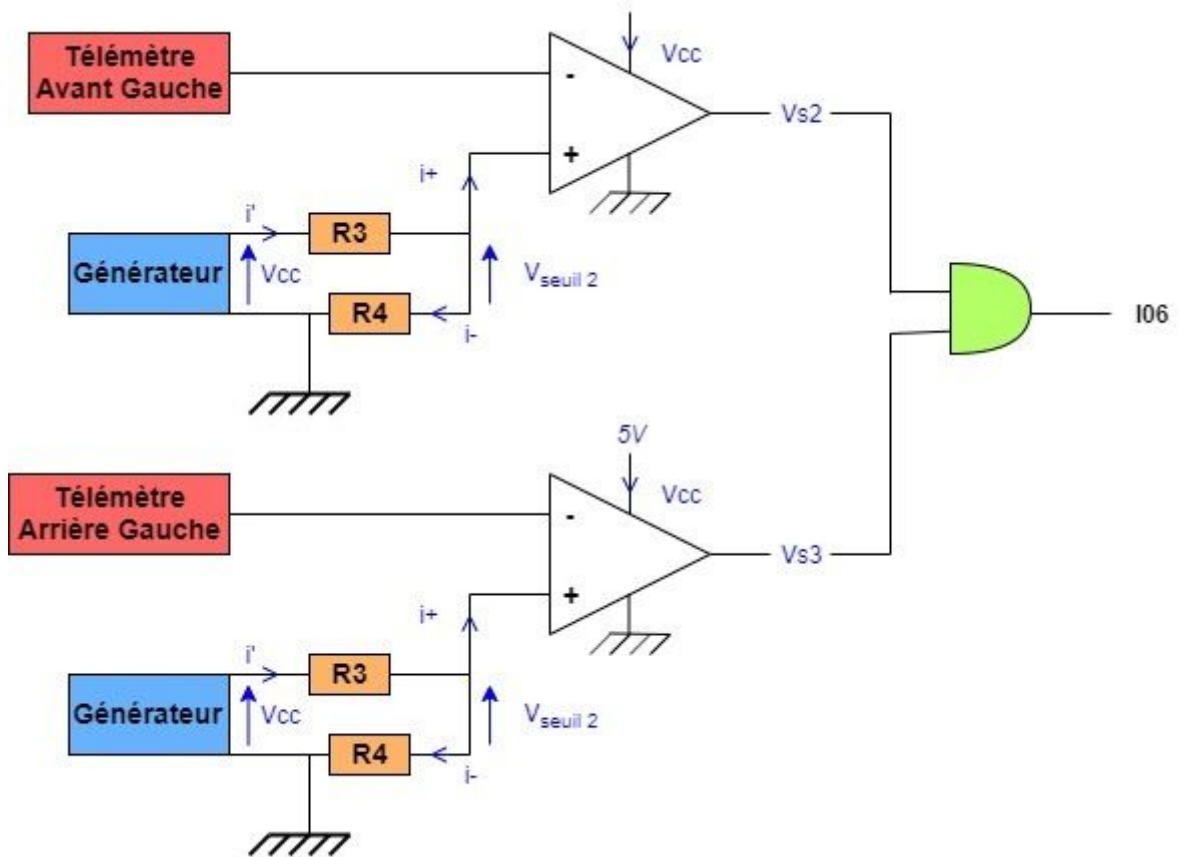


Le choix des résistances a été fait en prenant le meilleur ordre de grandeur possible, pour éviter qu'elle soit trop faible (consommation de beaucoup de puissance) ou qu'elle soit trop grande (augmentation du champ magnétique).

Nous avons utilisé lors de la conception du robot le comparateur de tension suivant, le TLV2464CN. Il a fallu faire très attention sur quelles entrées du microcontrôleur, les décisions du comparateur étaient renvoyées, afin de vérifier les branchements.



La conception de la détection de porte a été pensée de la même manière que celui précédemment avec quelques changements. En effet, pour un soucis de simplification de code pour la suite, l'ajout d'une fonction ET a été faite. De cette manière, la porte est détectée seulement si le télémètre avant gauche et le télémètre arrière gauche détectent une distance supérieure à 26cm. Ainsi, $V_{\text{seuil}2} = 0,6 \text{ V}$.



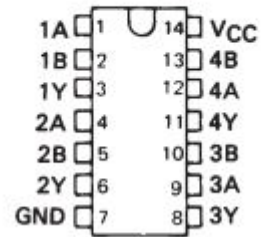
Dans cette configuration lorsque :

$V_{\text{seuil}2} > V_T \Rightarrow 5\text{V}$ (Niveau Haut) / Présence porte

$V_{\text{seuil}2} < V_T \Rightarrow 0\text{V}$ (Niveau Bas) / Absence porte

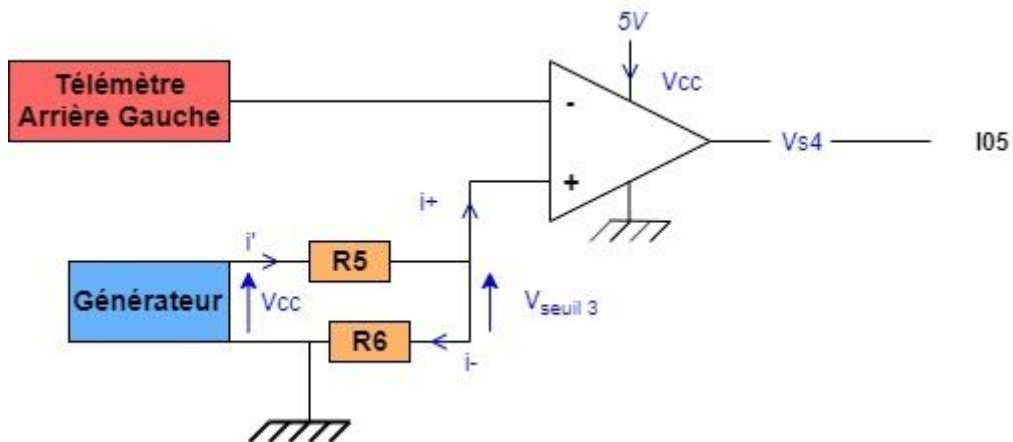
L'utilisation de ce dispositif SN74LS08N a engendré une petite contrainte. En effet, le branchement sur le comparateur de tension était primordial. Si nous inversions V_+ et V_- , alors la détection de la porte ne pouvait pas fonctionner. En effet, une partie de l'information étant triée au préalable, il fallait faire attention à quoi correspondait le niveau bas.

SN5408, SN54LS08, SN54S08 . . . J OR W PACKAGE
 SN7408 . . . J OR N PACKAGE
 SN74LS08, SN74S08 . . . D, J OR N PACKAGE
 (TOP VIEW)

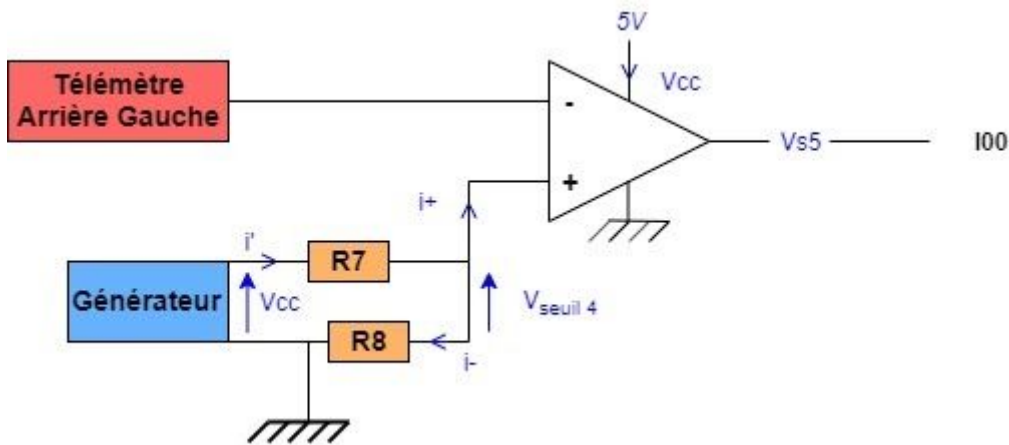


Pour la suite de l'interface, il a fallu reconnaître que notre robot n'avait pas une précision ultime et qu'il était possible qu'il ne tourne pas de 90° tout juste. Dans ce cas, la déviation de la trajectoire même légère peut contraindre le robot à toucher les parois. Pour éviter ceci, nous avons créé un asservissement des distances latérales.

Nous avons alors considéré la décision Mur Trop Près et Mur Trop Loin. Celles-ci correspondent respectivement à une distance au mur inférieure à 6cm et à une distance au mur supérieure à 10cm.



Par ailleurs, nous avons décidé de mettre l'asservissement sur le télémètre arrière, afin que le robot ne confonde pas le début de la porte avec la décision robot trop loin.



Cette asservissement permet au robot de gagner en précision et en efficacité.

V - Conclusion

L'interface labyrinthe utilisant alors cinq comparateurs de tension, transmet alors 4 décisions indépendantes au microcontrôleur. Il faut alors que chaque entrée du microcontrôleur soit parfaitement identifiée. De cette manière, il est primordial que les personnes s'occupant des deux interfaces se mettent d'accord sur la manière dont ils voient le fonctionnement, afin d'éviter une grosse perte de temps.

4.5. L'interface de traitement de distances des roues

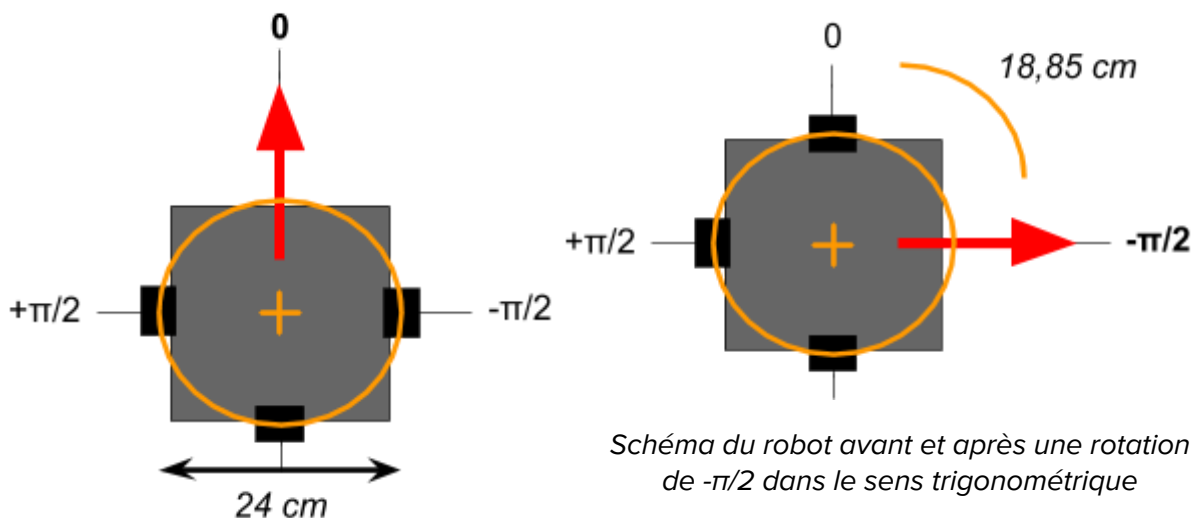
Suivant les différentes étapes de franchissement du labyrinthe, les roues sont amenées à tourner un certain temps, dans certaines directions... Afin de minimiser la place utilisée par le robot lors de ses rotations, les roues ont été placées au centre des côtés de la plaque métallique et non pas près des bords, puisque la distance entre deux murs intérieurs du labyrinthe peut être de seulement 40 cm. De ce fait, et sous réserve d'un bon placement central du robot dans les couloirs du labyrinthe, on parvient à respecter la fonction contrainte consistant à ne pas en toucher les murs.

Étapes nécessitant une rotation particulière des roues :

1. Pivot en surplace pour l'alignement de départ ;
2. Rotation de $-(\pi/2)$ une fois entré dans le labyrinthe ;
3. Rotation de $+(\pi/2)$ une fois la porte intermédiaire atteinte ;
4. Rotation $-(\pi/2)$ une fois la porte intermédiaire franchie ;
5. Rotation de $+(\pi/2)$ une fois la porte de sortie atteinte.

L'étape 1 est celle qui a été la plus simple à mettre en place. En effet, les roues étant placées au centre d'inertie du robot, il suffit de les faire tourner simultanément à la même vitesse, mais dans des sens opposés afin d'entraîner un pivotement du robot sur son point de départ.

Les étapes suivantes ont elles-aussi été mises en place sur ce principe, à la différence près que nous connaissions l'angle de rotation ($\pm \pi/2$) et avons besoin de le respecter au mieux possible, elles ont donc nécessité quelques calculs et réflexions supplémentaires...



Nous avons alors besoin de faire le lien entre la rotation d'une roue ($D=7,8$ cm) et l'angle de rotation du robot, sachant que l'axe des roues mesure $L=24$ cm. Combien de tours de roues faut-il faire pour pivoter de $\pm(\pi/2)$?... Un angle de $\pi/2$ correspond à un quart de tour, on peut donc calculer la distance parcourue par le robot lorsqu'il effectue un quart de tour, à savoir $\frac{1}{4}$ de son périmètre de rotation :

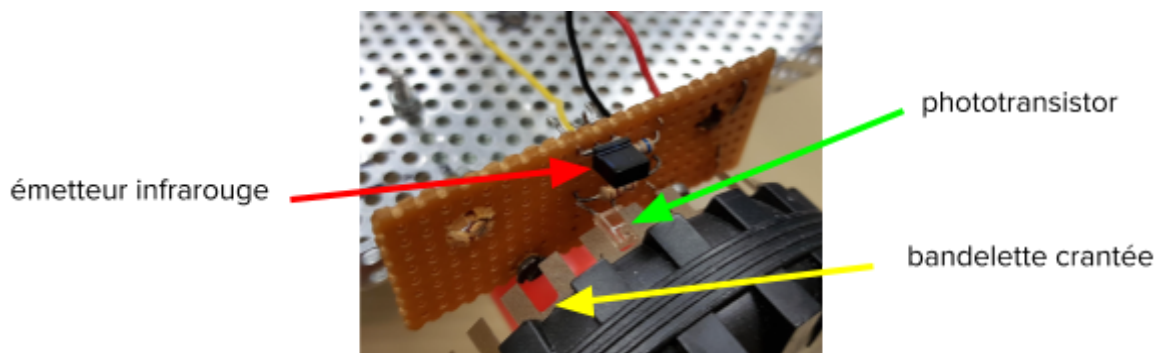
$$P = \pi \times L = 24\pi$$

$$\Rightarrow d = \frac{1}{4} \times P = \frac{1}{4} \times 24\pi = 6\pi \approx 18,85 \text{ cm}$$

En connaissant la distance à parcourir par une roue du robot pour qu'il pivote de $\pm(\pi/2)$, il nous suffit maintenant de mesurer la distance parcourue par une roue lors d'un pivot, afin de savoir quand indiquer la fin du pivotement.

Pour ce faire, nous avons choisi d'utiliser un phototransistor et son émetteur, tous deux à notre disposition dans le cadre de ce projet. Nous avons ainsi monté une bandelette constituée de 37 crans tout autour de l'une des deux roues. Le phototransistor a lui été soudé sur une plaque avec son émetteur infrarouge.

L'ensemble a été placé à la manière de la photo ci-dessous, de telle sorte que lorsque la roue tourne, les crans cachent, puis laissent passer la lumière infrarouge. Grâce au montage de résistances réalisé sur la plaque qui sert de support, le phototransistor renvoie un signal binaire en sortie : 1 lorsqu'il reçoit le rayon de l'émetteur, 0 dans le cas contraire.



La bandelette est constituée de 37 crans, correspondant donc à un tour de roue. De plus, on peut calculer le périmètre d'une roue de diamètre 7,8cm:

$$\Rightarrow Pr = \pi \times D = 7,8 \pi \approx 24,50 \text{ cm}$$

Calculons donc le nombre de tours à faire, connaissant la distance d à parcourir par la roue :

$$\Rightarrow N = d / Pr = 18,85 / 24,50 \approx 0,79 \text{ tour}$$

Or 1 tour correspond à 37 picots, donc 0,79 tour équivaut à :

$$0,79 \times 37 \approx 29 \text{ picots}$$

Nous avons ainsi calculé le nombre de passages (29) entre les valeurs binaires 0 et 1, renvoyées par le phototransistor, qu'il est nécessaire de comptabiliser au sein du programme avant de pouvoir affirmer (théoriquement) qu'un pivotement de $\pm (\pi/2)$ a été effectué.

4.6. L'interface de gestion de l'énergie

Dans le cadre de ce projet, la gestion de l'énergie n'est pas demandée. Nous aurons à notre disposition un générateur auquel notre robot sera branché.

Nous avons juste ajouté un interrupteur lié à la carte micro PicBasic et au générateur permettant de contrôler facilement la mise en marche de notre robot.

5. ÉTAPE DE RÉALISATION : ASSEMBLAGE DU ROBOT

Après avoir étudié toutes les interfaces et réalisé tous les tests permettant leur bon fonctionnement, il a fallu assembler le robot.

Nous disposons d'une plateforme métallique permettant de placer tous les éléments nécessaires au fonctionnement du robot, d'un interrupteur qui sera lié au générateur, de deux moteurs liés à des roues permettant au robot de se déplacer ainsi que d'une troisième roue qui va permettre d'assurer la stabilité du robot.

Le robot doit être stable donc il a donc fallu placer les différents composants de telle sorte que le robot ne penche pas d'un côté ou de l'autre.

Pour cela, nous avons d'abord placé les deux moteurs de chaque côté de la plaque (à peu près à la même distance de l'avant et l'arrière de la plaque) ainsi qu'une troisième roue à l'arrière pour la stabilité.

La grande carte PicBasic est placée à l'arrière de la plaque pour éviter que le robot ne bascule vers l'avant lorsqu'il est en marche.

Les deux ensembles "phototransistor et émetteur" sont placés face aux roues (Cf : partie 4.5: interface de traitement de distance des roues).

Les interfaces lampe et labyrinthe se trouvent sur la plaque d'essai se situant à l'avant de notre robot. Les deux photorésistances ont été placés, et isolés, sur une "tête" fabriqué en carton par nos soins à environ 30cm au-dessus de la plateforme. Cette "tête" est portée par une tige en plastique de cette même longueur. Les trois télémètres ont été fixés sur les côtés de la plateforme métallique : un à l'avant et les deux autres sur le côté gauche du robot.

6. CONCLUSION

Nous avons eu onze séances pour finaliser notre projet dont une servant au concours du robot et à la soutenance orale. La réalisation du projet était une nouveauté pour l'ensemble du groupe, c'est pour cette raison que nous avons de nombreuses choses à améliorer, tels qu'une meilleure organisation en passant par des diagrammes de Gantt. Par ailleurs, nous devons également prendre le temps à la fin et au début de chaque séance pour canaliser les énergies sur des points précis de l'avancement du robot. De plus, une meilleure mise au travail aurait permis un avancement nettement plus rapide et des objectifs tenus. Le projet robot a été une expérience non-négligeable dans le cadre d'une formation d'ingénieur malgré de nombreuses difficultés rencontrées, mais peuvent être corrigées grâce à cette rétrospective. S'il y avait un bilan à donner, c'est que même si le projet n'a pas complètement abouti, nous avons pris beaucoup d'expérience suite à nos erreurs.

BIBLIOGRAPHIE

Datasheet TLV2464CN, <https://www.ti.com/lit/gpn/tlv2460>

Datasheet SHARP 2D120,
<https://www.pololu.com/file/OJ157/GP2D120-DATA-SHEET.pdf>

Datasheet SN74LS08N, <https://www.ti.com/lit/pdf/sdls033>

Manuel PicBasic,
http://www.polytech.univ-nantes.fr/etn/documents/Manuel_PicBasic_USB2011.pdf